

# On the Placement Delivery Array Design in Centralized Coded Caching Scheme

Qifa Yan, Minquan Cheng, Xiaohu Tang, and Qingchun Chen

## Abstract

Caching is a promising solution to satisfy the ever-increasing demands for the multi-media traffics. In caching networks, coded caching is a recently proposed technique that achieves significant performance gains over the uncoded caching schemes. However, to implement the coded caching schemes, each file has to be split into  $F$  packets, which usually increases exponentially with the number of users  $K$ . Thus, designing caching schemes that decrease the order of  $F$  is meaningful for practical implementations. In this paper, by reviewing the Ali-Niesen caching scheme, the placement delivery array (PDA) design problem is firstly formulated to characterize the placement issue and the delivery issue with a single array. Moreover, it is disclosed that the problem of designing a centralized coded caching scheme can be translated into a problem of designing an appropriate PDA. Secondly, it is shown that the Ali-Niesen scheme corresponds to a special class of PDA, which realizes the best coding gain with the least  $F$ . Thirdly, we present a new construction of PDA for the centralized caching system, wherein the cache size of each user (identical cache size is assumed at all users) is a fraction  $1/q$  or  $(q-1)/q$  ( $q$  is an integer such that  $q \geq 2$ ) of the server. The new construction can decrease the required  $F$  from the order  $O\left(e^{K \cdot \left(\frac{1}{q} \ln q + (1-\frac{1}{q}) \ln \frac{q}{q-1}\right)}\right)$  of Ali-Niesen scheme to  $O\left(e^{K \cdot \frac{1}{q} \ln q}\right)$ , which saves a factor of  $O\left(e^{K \cdot (1-\frac{1}{q}) \ln \frac{q}{q-1}}\right)$  that increases exponentially with  $K$ , while the coding gain loss is only 1.

## Index Terms

Placement Delivery Array, Centralized Coded Caching, Content Delivery Network.

## I. INTRODUCTION

Driven by the broadband services such as video-on-demand (VoD) and catch-up TV, wireless data traffic is predicted to increase dramatically in the next few years, up to two orders of magnitude by 2020 [3]. However, it is widely acknowledged that the current wireless architecture can not effectively support the ever-increasing traffic, in particular when the allocated spectrum resource is limited, which consequentially leads to congestion in peak times. Fortunately, there is a common feature of asynchronous content reuse in video streaming applications, *i.e.*, the same contents are requested by different users at different times [10]. As a result, one promising approach to alleviate the congestion is “removing” some traffic delivery by disseminating contents into memories across the network when the network load is low. This dissemination is referred to as *content placement* or *content caching*. During the content delivery at peak times, different users’ requests can be better satisfied with the help of content caching. Therefore, the congestion can be effectively alleviated through two separate phases, namely, the *content placement phase* at off peak times and the *content delivery phase* at peak times, respectively.

Previous researches focused on either content placement or content delivery design. For example, different content placements and content cache location choices were studied in [2], [11] to improve the quality-of-service. The allocations of the replicated contents in the caches across the network were investigated in [4], [9] to reduce the average access cost. While the multicast gains to users with the similar demands were addressed in [1], [5]. The key idea of these conventional caching design is to deliver part of the content to caches close to the end users, so that the requested content can be served locally. Recently in the seminal work of coded caching scheme in [7], Maddah-Ali and Niesen had disclosed that better coding gain can be achieved

Q.F. Yan, X.H. Tang, and Q.C. Chen are with The School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China, 610031. E-mails: qifa@my.swjtu.edu.cn, xhutang@swjtu.edu.cn, qcchen@swjtu.edu.cn.

M.Q. Cheng is with The School of Computer Science and Information Technology, Guangxi Normal University, Guilin, China, 541004. E-mail: chengqinshi@hotmail.com.

by exploiting caches to create multicast opportunities. Indeed, the system studied in [7] is a centralized caching scheme, where a central server coordinates all the transmissions. By jointly designing the content placement phase and the content delivery phase, the central sever is able to simultaneously deliver distinct contents to different users through a shared link. In order to simplify the design, Ali and Niesen adopted the following two strategies:

- S1. In the content placement phase, identical caching policy is assumed, *i.e.*, every user will cache exactly the *same* packets of all files;
- S2. In the content delivery phase, the requested packets by users will be XOR multiplexing to formulate the delivered signal.

It is shown that, with an elaborate caching design in content placement phase and the XOR multiplexing of those requested packets in content delivery phase, the near-optimal scheme can be realized for arbitrary traffic demands by using the Ali-Niesen scheme in [7]. Nevertheless, in general, one disadvantage of Ali-Niesen scheme is that, it usually needs to split each file into  $F$  packets, where  $F$  increases exponentially with the number of users  $K$ . This would become infeasible when  $K$  is properly large. Therefore, designing a coded caching scheme with smaller size  $F$  will be a critical issue, especially for practical implementations.

In this paper, just like the Ali-Niesen scheme, we assume the same two strategies in the coded caching scheme design as well. However, unlike the previous analysis, firstly, we propose to use *placement delivery array (PDA)* to characterize the involved strategies in two phases. Essentially, PDA is an  $F \times K$  array consisting of a specific symbol “\*” and some integers, wherein the row index  $j$  ( $j \in \{0, 1, \dots, F-1\}$ ) and the column index  $k$  ( $k \in \{0, 1, \dots, K-1\}$ ) stands for the  $j$ -th packets of all the files for the  $k$ -th user. In particular, a PDA can depict the following coded caching scheme

1. In the content placement phase, the symbol “\*” at row  $j$  and column  $k$  in PDA indicates that user  $k$  stores the  $j$ -th packet of all the files;
2. In the content delivery phase, the request packets by different users, which are indicated by the same integers at each row, will be sent by the sever simultaneously after XORing operation.

In this way, we can depict the placement and delivery schemes for all possible requests in a single array. As a result, the problem of designing a centralized coded caching scheme can be transformed into the problem of designing an appropriate PDA. In fact, Ali-Niesen scheme corresponds to the so-called regular PDA, where the occurrence of each integer is a constant. Notably, we establish an upper bound on the achieved coding gain of the regular PDA, which reveals that Ali-Niesen scheme achieves the upper bound with the least  $F$ . Unfortunately,  $F$  will be increased exponentially with the number of users  $K$ . Thus, designing caching schemes that decrease the order of  $F$  is meaningful for practical implementations. In this paper, we aim at the centralized coded caching scheme design with reduced requirement in the size of  $F$ . More specifically, we present a new construction of PDA, wherein the cache size of each user (identical cache size is assumed at all users) is a fraction  $1/q$  or  $(q-1)/q$  ( $q$  is an integer *s.t.*  $q \geq 2$ ) of the server. The new construction is able to significantly decrease the required  $F$  from the order  $O\left(e^{K \cdot (\frac{1}{q} \ln q + (1-\frac{1}{q}) \ln \frac{q}{q-1})}\right)$  of Ali-Niesen scheme to  $O\left(e^{K \cdot \frac{1}{q} \ln q}\right)$ , which corresponds to a saving of  $O\left(e^{K \cdot (1-\frac{1}{q}) \ln \frac{q}{q-1}}\right)$  in  $F$  that increases exponentially with  $K$ , while the coding gain loss is only 1.

The remainder of this paper is organized as follows. In Section II, the system model and Ali-Niesen scheme are briefly reviewed. In Section III, the definition of PDA is introduced and then its connection with coded caching scheme is established. In Section IV, the Ali-Niesen scheme is re-explained by using the regular PDA and its optimality is proved in terms of its capability to approach the upper bound on the coding gain for regular PDA. In Sections V and VI, a new PDA construction and its performance comparison with the Ali-Niesen scheme are presented, respectively. Finally, the conclusion is given in Section VII.

**Notations:** In this paper, arrays are denoted by bold capital letters, vectors are denoted by bold lower case letters. We use  $[i, j]$  to denote the set of integers  $\{i, i+1, \dots, j\}$  and  $[i, j)$  to denote the set  $\{i, i+1, \dots, j-1\}$ . For two series  $\{a_n\}, \{b_n\}$ ,  $a_n \sim b_n$  means  $\lim_{n \rightarrow \infty} \frac{a_n}{b_n} = 1$ . The operation  $\oplus$  means bitwise Exclusive OR (XOR) operation of two packets. The set of positive integers is denoted by  $\mathbb{N}^+$ .

## II. NETWORK MODEL AND ALI-NIESEN SCHEME

Let us consider a caching system consisting of one server, connected by  $K$  users through an error-free shared link. The server has  $N$  files ( $N \geq K$ ), which are denoted by  $\mathcal{W} = \{W_0, W_1, \dots, W_{N-1}\}$ . Without loss of generality (W.L.O.G.), we assume that each file is of unit length. Denote the  $K$  users by  $\mathcal{K} = \{0, 1, \dots, K-1\}$ , each having a cache of size  $M$  units, where  $0 \leq M \leq N$ . Fig. 1 shows a diagram of the aforementioned caching system.

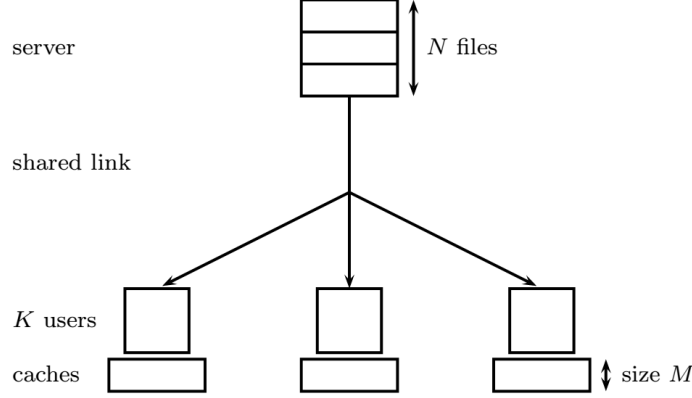


Fig. 1: Caching system

The caching system operates in two separated phases:

1. In the placement phase, a file is sub-divided into  $F$  equal packets<sup>1</sup>, *i.e.*,  $W_i = \{W_{i,j} : j \in [0, F)\}$ , each of size  $1/F$  units. Then, each packet is placed in different user caches deterministically. Denote the contents at user  $k$  by  $\mathbf{Z}_k$ , where  $k \in \mathcal{K}$ .
2. In the delivery phase, each user randomly requests one file from the files set  $\mathcal{W}$  independently. The request is denoted by  $\mathbf{d} = (d_0, d_1, \dots, d_{K-1})$ , which indicates that user  $k$  requests the  $d_k$ -th file  $W_{d_k}$  for any  $d_k \in [0, N)$  and  $k \in \mathcal{K}$ . Once the server received the users' request  $\mathbf{d}$ , it broadcasts a signal of at most  $S_{\mathbf{d}}$  packets to users. Each user is able to recover its requested file from the signal received in the delivery phase with the help of the contents within its own cache.

The caching system is parameterized by  $K, M, N$ , so in this paper we call it a  $(K, M, N)$  caching system. Following the convention, we refer to a realization of placement and delivery as a caching scheme. In a caching scheme, if each file is sub-divided into  $F$  packets, we refer to such a scheme as a  $F$ -division caching scheme. Specifically, we define the rate of the  $F$ -division scheme as

$$R = \sup_{\substack{\mathbf{d} = (d_0, \dots, d_{K-1}) \\ d_k \in [0, N), \forall k \in [0, K)}} \left\{ \frac{S_{\mathbf{d}}}{F} \right\}. \quad (1)$$

The definition can be explained as follows: Assume that in the placement phase, the server constructs a code for each possible request. During the delivery phase, the server responds the users' request  $\mathbf{d}$  by sending the corresponding code of length  $S_{\mathbf{d}}/F$ . In the sense of source coding,  $R$  is the worst coding rate (normalized by the file size) over all possible requests  $\mathbf{d}$ . Then  $1/R$  (up to a constant factor) yields the minimal throughput for the shared link in order to satisfy all users for any request [8]. That is, the smaller  $R$ , the less load at the server. Therefore, the primary concern for a given  $(K, M, N)$  caching system is to minimize the rate  $R$ .

In [7], Ali and Niesen proposed a coded scheme for the  $(K, M, N)$  caching system under the assumption that the files can be arbitrarily divided. Algorithm 1 depicts a  $\binom{K}{KM/N}$ -division Ali-Niesen scheme where  $M/N \in \{0, 1/K, 2/K, \dots, 1\}$ . It

<sup>1</sup> Memory sharing technique may lead to non equally divided packets [7], in this paper, we will not discuss this case.

was shown in [7] that, the scheme is feasible and each user can successfully recover its requested file at a rate

$$R_{A-N} \left( \frac{M}{N}, K \right) = K \left( 1 - \frac{M}{N} \right) \cdot \frac{1}{1 + \frac{KM}{N}} \quad (2)$$

for  $M/N \in \{0, 1/K, 2/K, \dots, 1\}$ . For general  $0 \leq M/N \leq 1$ , the lower convex envelop of these points can be achieved by memory sharing technique [7].

---

**Algorithm 1** Ali-Niesen Caching Scheme

---

```

1: procedure PLACEMENT( $W_0, W_1, \dots, W_{N-1}$ )
2:    $t \leftarrow \frac{KM}{N}$ 
3:    $\mathfrak{T} \leftarrow \{\mathcal{T} \subset \mathcal{K} : |\mathcal{T}| = t\}$ 
4:   for  $n \in [0, N)$  do
5:     Split  $W_n$  into  $\{W_{n,\mathcal{T}} : \mathcal{T} \in \mathfrak{T}\}$  of equal packets
6:   end for
7:   for  $k \in \mathcal{K}$  do
8:      $\mathbf{Z}_k \leftarrow \{W_{n,\mathcal{T}} : n \in [0, N), \mathcal{T} \in \mathfrak{T}, k \in \mathcal{T}\}$ 
9:   end for
10: end procedure
11: procedure DELIVERY( $W_0, W_1, \dots, W_{N-1}, d_0, d_1, \dots, d_{K-1}$ )
12:    $t \leftarrow \frac{KM}{N}$ 
13:    $\mathfrak{S} \leftarrow \{\mathcal{S} \subset \mathcal{K} : |\mathcal{S}| = t + 1\}$ 
14:   Server sends  $\{\oplus_{k \in \mathcal{S}} W_{d_k, \mathcal{S} \setminus \{k\}} : \mathcal{S} \in \mathfrak{S}\}$ 
15: end procedure

```

---

For clarity, we trim an example from [7] to illustrate the scheme.

*Example 1.* Consider the case  $N = K = 2$ , *i.e.*, there are two files, say  $W_0 = A$ ,  $W_1 = B$  and two users each with cache memory size  $M = 1$ . According to Algorithm 1, we have  $t = KM/N = 1$ . Then  $A, B$  are partitioned into  $\binom{K}{t} = \binom{2}{1} = 2$  packets of equal size, *i.e.*  $A = \{A_{\{0\}}, A_{\{1\}}\}$  and  $B = \{B_{\{0\}}, B_{\{1\}}\}$ . By simplicity, we abbreviate  $\{0\}$  and  $\{1\}$  as 0 and 1, respectively. That is,  $A = \{A_0, A_1\}$  and  $B = \{B_0, B_1\}$ . In the placement phase, user 0 caches  $\mathbf{Z}_0 = \{A_0, B_0\}$ , while user 1 caches  $\mathbf{Z}_1 = \{A_1, B_1\}$ . In the delivery phase, assume that user 0 requests  $A$  and user 1 requests  $B$ . Since user 0 has packet  $A_0$  of  $A$ , it only needs to obtain the missing packet  $A_1$ , which is cached by user 1. Similarly, user 1 needs only to obtain the packet  $B_0$  which is in the cache of user 0. Thus, the server can simply send  $A_1 \oplus B_0$ . Since user 0 has  $B_0$  in its cache, it can decode  $A_1$  from the signal  $A_1 \oplus B_0$ . Similarly, user 1 can decode its missing packet  $B_1$ . The signals sent for other three requests are illustrated in Fig. 2. It is easy to see that this scheme achieves a rate  $1/2$  since each packet  $A_0, A_1, B_0, B_1$  is of length  $1/2$  unit, so does their XOR values.

Nevertheless, for the same  $(K, M, N)$  caching system, the conventional non-coded scheme has rate

$$R_{\text{Conventional}} \left( \frac{M}{N}, K \right) = K \left( 1 - \frac{M}{N} \right) \quad (3)$$

Compare (2) with (3), it is seen that the gain for convention non-coded scheme is relevant to the the ratio of local cache memory  $M$  to the total content  $N$ , whereas a new coding gain for Ali-Niesen scheme comes from aggregate global cache size  $KM$  through XOR coding, even though there is no cooperation among the users.

### III. PLACEMENT DELIVERY ARRAY

In this section, we propose a new concept to characterize the caching system.

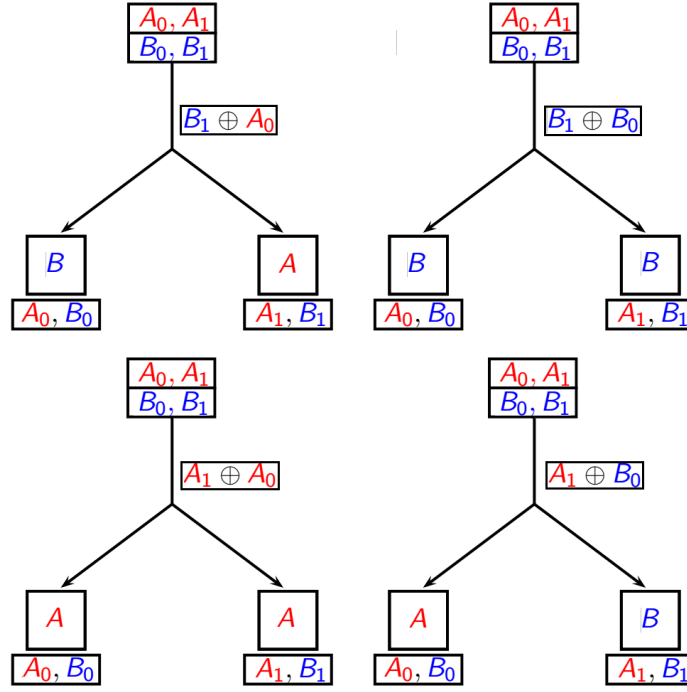


Fig. 2: Caching scheme for  $N = K = 2, M = 1$  with all four possible user requests.

**Definition 1.** For positive integers  $K, F, Z$  and  $S$ , a  $F \times K$  array  $\mathbf{P} = [p_{j,k}]$ ,  $j \in [0, F), k \in [0, K)$ , composed of a specific symbol “\*” and  $S$  nonnegative integers  $0, 1, \dots, S-1$ , is called a  $(K, F, Z, S)$  placement delivery array (PDA) if it satisfies the following conditions:

- C1. The symbol “\*” appears  $Z$  times in each column;
- C2. Each integer occurs at least once in the array;
- C3. For any two distinct entries  $p_{j_1, k_1}$  and  $p_{j_2, k_2}$ ,  $p_{j_1, k_1} = p_{j_2, k_2} = s$  is an integer only if
  - a.  $j_1 \neq j_2, k_1 \neq k_2$ , i.e., they lie in distinct rows and distinct columns; and
  - b.  $p_{j_1, k_2} = p_{j_2, k_1} = *$ , i.e., the corresponding  $2 \times 2$  subarray formed by rows  $j_1, j_2$  and columns  $k_1, k_2$  must be of the following form

$$\begin{bmatrix} s & * \\ * & s \end{bmatrix} \text{ or } \begin{bmatrix} * & s \\ s & * \end{bmatrix}$$

Based on a  $(K, F, Z, S)$  PDA  $\mathbf{P} = [p_{j,k}]$  with  $j \in [0, F)$  and  $k \in [0, K)$ , a  $F$ -division caching scheme for a  $(K, M, N)$  caching system with  $Z/F = M/N$  can be conducted as follows:

1. **Placement Phase:** All the files are cached in the same manner. Each file  $W_i$  is split into  $F$  packets, i.e.  $W_i = \{W_{i,j} : j \in [0, F)\}$ ,  $\forall i \in [0, N)$ , so that user  $k \in \mathcal{K}$  caches packets

$$\mathbf{Z}_k = \{W_{i,j} : p_{j,k} = *, \forall i \in [0, N)\} \quad (4)$$

By C1, each user stores  $N \cdot Z$  packets. Since each packet has size  $1/F$ , the whole size of cache is  $N \cdot Z \cdot 1/F = M$ , which satisfies the users' cache constraint.

2. **Delivery Phase:** Once the server receives the request  $\mathbf{d} = (d_0, d_1, \dots, d_{K-1})$ , at the time slot  $s$ ,  $0 \leq s < S$ , it sends

$$\bigoplus_{p_{j,k}=s, j \in [0, F), k \in [0, K)} W_{d_k, j} \quad (5)$$

Assume that in PDA  $\mathbf{P}$  there are  $t$  entries  $p_{j_1, k_1} = p_{j_2, k_2} = \dots = p_{j_t, k_t} = s$  where  $0 \leq j_1, \dots, j_t < F$  and  $0 \leq k_1, \dots, k_t < K$ . Consider the subarray formed by rows  $j_1, \dots, j_t$  and columns  $k_1, \dots, k_t$ , which is of order  $t \times t$  since

$j_u \neq j_v$  and  $k_u \neq k_v$  for all  $1 \leq u \neq v \leq t$  by C3-a. Further, applying C3-b we have  $p_{j_u, k_v} = *$  for all  $1 \leq u \neq v \leq t$ . That is to say, this subarray is equivalent to the following  $t \times t$  array

$$\begin{bmatrix} s & * & \cdots & * \\ * & s & \cdots & * \\ \vdots & \vdots & \ddots & \vdots \\ * & * & \cdots & s \end{bmatrix} \quad (6)$$

with respect to row/column permutation. According to (5), at the time slot  $s$ ,  $0 \leq s < S$ , the sever sends

$$\bigoplus_{1 \leq u \leq t} W_{d_{k_u}, j_u}$$

Note from (6) that in column  $v$ , all the entries are “\*”s except for the  $v$ th one. Then it follows (4) that user  $k_v$  has already all the other packets  $W_{d_{k_u}, j_u}$ ,  $1 \leq u \neq v \leq t$ , in its cache at the placement phase. Then, it can easily decode the desired packet  $W_{d_{k_v}, j_v}$ . Since the server sends  $S$  packets for each possible request  $\mathbf{d}$ , the rate of the scheme is given by  $S/F$ .

By the above analysis, we have proved the following theorem.

**Theorem 1.** *A  $F$ -division caching scheme for a  $(K, M, N)$  caching system can be realized by a  $(K, F, Z, S)$  PDA  $\mathbf{P} = [p_{j,k}]_{F \times K}$  with  $Z/F = M/N$ . Precisely, each user is able to decode its requested file correctly for any request  $\mathbf{d}$  at the rate  $R = S/F$ .*

Theorem 1 establishes the connection between a  $(K, F, Z, S)$  PDA and a  $F$ -division caching scheme for a  $(K, M, N)$  caching system, which suggests a way to transform the design of a caching scheme into a construction of appropriate PDA. Actually, PDA can be used to describe a subclass of caching schemes employing strategies S1 and S2, which includes Ali-Niesen scheme. For instance, the scheme in Example 1 can be described by a  $(2, 2, 1, 1)$  PDA

$$\begin{bmatrix} * & 0 \\ 0 & * \end{bmatrix}$$

In the next section, we will rebuild Ali-Niesen scheme by means of PDA in detail.

Finally, we conclude this section by an illustrative example.

*Example 2.* It is easily checked that the array

$$\mathbf{A}^{2,2} = \begin{bmatrix} * & 1 & * & 2 & * & 0 \\ 0 & * & * & 3 & 1 & * \\ * & 3 & 0 & * & 2 & * \\ 2 & * & 1 & * & * & 3 \end{bmatrix} \quad (7)$$

is a  $(6, 4, 2, 4)$  PDA.

Assume that in a  $(6, 3, 6)$  caching system, the files are denoted by  $W_0, W_1, W_2, W_3, W_4, W_5$  respectively, and each file is divided into  $F = 4$  packets, i.e.  $W_i = \{W_{i,0}, W_{i,1}, W_{i,2}, W_{i,3}\}$ ,  $i \in [0, 6)$ . This system can be implemented according to  $\mathbf{A}^{2,2}$  as follows:

- **Placement Phase:** The contents in each users are

$$\begin{aligned} \mathbf{Z}_0 &= \{W_{i,0}, W_{i,2} : i \in [0, 6)\} & \mathbf{Z}_1 &= \{W_{i,1}, W_{i,3} : i \in [0, 6)\} \\ \mathbf{Z}_2 &= \{W_{i,0}, W_{i,1} : i \in [0, 6)\} & \mathbf{Z}_3 &= \{W_{i,2}, W_{i,3} : i \in [0, 6)\} \\ \mathbf{Z}_4 &= \{W_{i,0}, W_{i,3} : i \in [0, 6)\} & \mathbf{Z}_5 &= \{W_{i,1}, W_{i,2} : i \in [0, 6)\} \end{aligned}$$

- **Delivery Phase:** Assume the request vector is  $\mathbf{d} = (0, 1, \dots, 5)$ . Table I shows the transmitting process.

TABLE I: Deliver steps in Example 2

| Time Slot | Transmitted Signal                      |
|-----------|---|
| 0         | $W_{0,1} \oplus W_{2,2} \oplus W_{5,0}$ |
| 1         | $W_{1,0} \oplus W_{2,3} \oplus W_{4,1}$ |
| 2         | $W_{0,3} \oplus W_{3,0} \oplus W_{4,2}$ |
| 3         | $W_{1,2} \oplus W_{3,1} \oplus W_{5,3}$ |

## IV. PDA FOR ALI-NIESEN SCHEME AND ITS OPTIMALITY

In this section, we firstly prove that Ali-Niesen scheme corresponds to a special class of PDA, regular PDA. Next, we show its optimality by establishing an upper bound on the coding gain for the regular PDA.

Recall that, in Ali-Niesen scheme, let  $t = KM/N \in [0, K]$ , then a file is split into  $\binom{K}{t}$  packets and each packet is labeled by a subset of size  $t$  in the set  $\mathcal{K} = \{0, 1, \dots, K-1\}$ . In the delivery phase, Algorithm 1 visits each subset of size  $t+1$  one by one. Let us arrange all such subsets in the lexicographic order and then define  $f_{t+1}(\mathcal{S})$  to be its order minus 1 for any subset  $\mathcal{S}$  of size  $t+1$ . Clearly,  $f_{t+1}$  is a bijection from  $\{\mathcal{S} \subset \mathcal{K} : |\mathcal{S}| = t+1\}$  to  $[0, \binom{K}{t+1})$ . For example, when  $K = 4$  and  $t = 1$ , in  $\mathcal{K} = \{0, 1, 2, 3\}$  all the subsets of size  $t+1 = 2$  are ordered as  $\{0, 1\}, \{0, 2\}, \{0, 3\}, \{1, 2\}, \{1, 3\}$ , and  $\{2, 3\}$ . Accordingly,  $f_2(\{0, 1\}) = 0, f_2(\{0, 2\}) = 1, f_2(\{0, 3\}) = 2, f_2(\{1, 2\}) = 3, f_2(\{1, 3\}) = 4$ , and  $f_2(\{2, 3\}) = 5$ .

Let  $\mathbf{D}^{K,t}$  be a  $\binom{K}{t} \times K$  array. Denote its rows by the sets in  $\{\mathcal{T} \subset \mathcal{K} : |\mathcal{T}| = t\}$ , and columns by  $0, 1, \dots, K-1$ , respectively. Then, define the entry  $d_{\mathcal{T},k}$  in row  $\mathcal{T}$  and column  $k$  as

$$d_{\mathcal{T},k} = \begin{cases} *, & \text{if } k \in \mathcal{T} \\ f_{t+1}(\mathcal{T} \cup \{k\}), & \text{if } k \notin \mathcal{T} \end{cases} \quad (8)$$

It is easily seen from Algorithm 1 that both the placement and delivery of Ali-Niesen scheme are essentially applications of  $\mathbf{D}^{K,t}$  to (4). Indeed,  $\mathbf{D}^{K,t}$  is a regular PDA.

**Definition 2.** An array  $\mathbf{P}$  is said to be a  $g$ -regular  $(K, F, Z, S)$  PDA,  $g$ -( $K, F, Z, S$ ) PDA or  $g$ -PDA for short, if it satisfies C1, C3, and

C2'. Each integer appears  $g$  times in  $\mathbf{P}$  where  $g$  is a constant.

In the corresponding caching scheme, a regular PDA leads to a fact that each packet sent is intended to  $g$  users. In what follows, we refer to  $g$  as *coding gain* for a  $g$ -( $K, F, Z, S$ ) PDA and its corresponding caching scheme. Obviously, the coding gain  $g$  is very desirable to be as large as possible. The following theorem shows that the Ali-Niesen scheme can be determined by a regular PDA with gain  $g = t+1$ .

**Theorem 2.** For a  $(K, M, N)$  caching system, such that  $M/N \in \{0, 1/K, 2/K, \dots, 1\}$ , let  $t = KM/N$ , then the array  $\mathbf{D}^{K,t}$  in (8) which corresponds to Ali-Niesen scheme, is a  $(t+1)$ -( $K, F, Z, S$ ) PDA, where  $F = \binom{K}{t}$ ,  $Z = \binom{K-1}{t-1}$ , and  $S = \binom{K}{t+1}$ .

*Proof:* From (8),  $\mathbf{D}^{K,t}$  is a  $\binom{K}{t} \times K$  array consisting of “\*” and integers in  $[0, \binom{K}{t+1})$ . Hereafter, it suffices to check C1, C2', and C3.

For each  $k \in \mathcal{K}$ ,  $k$  is included in exactly  $\binom{K-1}{t-1}$  subsets of  $\mathcal{K}$  in  $\{\mathcal{T} \subset \mathcal{K} : |\mathcal{T}| = t\}$ . Thus by (8), the symbol “\*” appears  $Z = \binom{K-1}{t-1}$  times in each column exactly. That is, C1 is satisfied.

Next, assume that two distinct entries  $d_{\mathcal{T}_1, k_1} = d_{\mathcal{T}_2, k_2} = s$  where  $\mathcal{T}_1, \mathcal{T}_2 \subset \mathcal{K}$  with  $|\mathcal{T}_1| = |\mathcal{T}_2| = t$  and  $k_1, k_2 \in \mathcal{K}$ . Then applying the fact that  $f_{t+1}$  is a bijection from  $\{\mathcal{S} \subset \mathcal{K} : |\mathcal{S}| = t+1\}$  to  $[0, \binom{K}{t+1})$ , from (8) we have that  $s$  is an integer if and only if

$$\mathcal{T}_1 \cup \{k_1\} = \mathcal{T}_2 \cup \{k_2\}$$

which implies that

- Each integer  $f_{t+1}(\mathcal{S})$  ( $|\mathcal{S}| = t + 1$ ) in  $[0, \binom{K}{t+1})$  occurs exactly  $t + 1$  times since there are  $t + 1$  distinct possibilities of  $(\{k_1\}, \mathcal{T}_1 = \mathcal{S} \setminus \{k_1\})$  with  $k_1$  ranging over  $\mathcal{S}$ ;
- $\mathcal{T}_1 \neq \mathcal{T}_2$  and  $k_1 \neq k_2$ , *i.e.* the two entries are in distinct rows and columns. Further, this is equivalent to  $k_1 \in \mathcal{T}_2, k_2 \in \mathcal{T}_1$ , and thus  $d_{\mathcal{T}_1, k_2} = d_{\mathcal{T}_2, k_1} = *$  by (8).

In other words, C2' and C3 hold. ■

*Example 3.* For a  $(6, 3, 6)$  caching system, Ali-Niesen scheme with  $t = 3$  can be depicted by the following PDA.

$$\mathbf{D}^{6,3} = \begin{bmatrix} * & * & * & 0 & 1 & 2 \\ * & * & 0 & * & 3 & 4 \\ * & * & 1 & 3 & * & 5 \\ * & * & 2 & 4 & 5 & * \\ * & 0 & * & * & 6 & 7 \\ * & 1 & * & 6 & * & 8 \\ * & 2 & * & 7 & 8 & * \\ * & 3 & 6 & * & * & 9 \\ * & 4 & 7 & * & 9 & * \\ * & 5 & 8 & 9 & * & * \\ 0 & * & * & * & 10 & 11 \\ 1 & * & * & 10 & * & 12 \\ 2 & * & * & 11 & 12 & * \\ 3 & * & 10 & * & * & 13 \\ 4 & * & 11 & * & 13 & * \\ 5 & * & 12 & 13 & * & * \\ 6 & 10 & * & * & * & 14 \\ 7 & 11 & * & * & 14 & * \\ 8 & 12 & * & 14 & * & * \\ 9 & 13 & 14 & * & * & * \end{bmatrix} \quad (9)$$

In the remainder of this section, we show the optimality of the PDA for Ali-Niesen scheme. Before that, we give two useful lemmas to characterize the relationship of parameters of a regular PDA.

**Lemma 1.** *For any given  $g$ -( $K, F, Z, S$ ) PDA, then the rate of the corresponding caching scheme is given by*

$$R = \frac{S}{F} = \frac{K(1 - \frac{Z}{F})}{g} \quad (10)$$

Moreover,  $g$  must satisfy

$$g \leq K \frac{Z}{F} + 1 \quad (11)$$

where the equality holds for  $g \geq 2$  only if each row has exactly  $KZ/F$  specific symbols “\*”.

*Proof:* To prove the rate given by (10), let us count the number of the integers in PDA in two different manners. On one hand, since each column has  $F - Z$  integers, there are totally  $K(F - Z)$  integers in all the  $K$  columns. On the other hand, since each integer occurs  $g$  times, the total number of all the  $S$  integers is  $Sg$ . Hence,

$$Sg = K(F - Z)$$

which results in (10).



To verify (11), denote  $t_j$  the number of symbol “\*” in row  $j$ ,  $j \in [0, F)$ . Since each integer occurs  $g$  times in the PDA, each row then contains at least  $g - 1$  “\*”s by C3-b, which immediately indicates  $t_j \geq g - 1$  for all  $j \in [0, F)$ . Note that totally there are  $KZ$  “\*”s. Thus, we have

$$F(g - 1) \leq \sum_{j=0}^{F-1} t_j = KZ$$

which gives (11) with equality holding only if each row has exactly  $g - 1 = KZ/F$  symbols “\*”.  $\blacksquare$

**Lemma 2.** *Given positive integers  $K, F$ , and  $g$  s.t.,  $K \geq g \geq 2$ , if a  $F \times K$  array  $\mathbf{P}$  whose entries consist of a specific symbol “\*” and some nonnegative integers satisfies C2', C3, and*

*C4. Each row has exactly  $g - 1$  “\*”s,*

*then  $F \geq \binom{K}{g-1}$ .*

*Proof:* We prove this lemma by the induction on the integer  $g \geq 2$ .

When  $g = 2$ , it follows from C4 that each row has one “\*” and  $K - 1$  integers. Note that a fact:

**Fact 1.** *The conditions C2', C3, and C4 are still satisfied after the exchange of two rows/columns in the  $F \times K$  array  $\mathbf{P}$ .*

So, we can always assume that the entry in the first row and the first column is “\*”. Then, the other ones in the first row are integers, which have to be  $K - 1$  distinct integers further by C3-a. Note that each of the  $K - 1$  integers occurs exactly  $g = 2$  times. Therefore by C3-b or (6), there are at least one “\*” in all the columns  $1, \dots, K - 1$ . Together with the “\*” in the first column, there are at least  $K$  “\*”s in this array. Since there is only one “\*” in each row, we conclude that, there are at least  $K$  rows, *i.e.*

$$F \geq K = \binom{K}{g-1}$$

holds for  $g = 2$  and all  $K \geq g$ .

Suppose that the claim holds for  $g = n$  and all  $K \geq g$ , *i.e.* for  $g = n$  and  $K \geq n$ , we have

$$F \geq \binom{K}{n-1} \quad (12)$$

if C2', C3, and C4 hold.

Let  $g = n + 1$  and  $K \geq n + 1$ . If a  $F \times K$  array  $\mathbf{P}$  satisfies C4, there are totally  $F(g - 1)$  “\*”s in  $\mathbf{P}$ . Then, in average, there are  $F(g - 1)/K$  “\*”s in each column. Thus, there exists a column having at most  $F(g - 1)/K$  “\*”s. Based on Fact 1, we can always transform the original array  $\mathbf{P}$  into another  $F \times K$  array  $\mathbf{P}'$  such that  $\mathbf{P}'$  is of form

$$\mathbf{P}' = \left[ \begin{array}{c|c} \begin{matrix} * \\ \vdots \\ * \end{matrix} & \mathbf{P}'_1 \\ \hline \begin{matrix} a_0 \\ \vdots \\ a_{m-1} \end{matrix} & \mathbf{P}'_2 \end{array} \right]$$

where  $a_0, \dots, a_{m-1}$  are integers,  $\mathbf{P}'_1$  and  $\mathbf{P}'_2$  are  $F_1 \times (K - 1)$  array and  $m \times (K - 1)$  array respectively, the integer  $m = F - F_1$ , and

$$F_1 \leq \frac{F(g - 1)}{K} = \frac{Fn}{K} \quad (13)$$

Denote the sets of integers in  $\mathbf{P}'_1$  and  $\mathbf{P}'_2$  by  $\mathcal{P}'_1$  and  $\mathcal{P}'_2$  respectively. Firstly, we know from C2' that  $a_j$  appears  $g - 1 \geq 1$  times in  $\mathcal{P}'_1 \cup \mathcal{P}'_2$  for any  $j \in [0, m)$ . Further, we have

$$\{a_0, \dots, a_{m-1}\} \cap \mathcal{P}'_2 = \emptyset \quad (14)$$

and thus

$$\{a_0, \dots, a_{m-1}\} \subseteq \mathcal{P}'_1 \quad (15)$$

since otherwise if  $a_j$  occurs in row  $j'$  of  $\mathbf{P}'_2$  for some integers  $j' \in [0, m)$ , then the element  $a_{j'}$  has to be “\*” by C3-b, a contradiction. Secondly, suppose that there is an integer  $b$  in the  $j$ th ( $j \in [0, F_1)$ ) row of  $\mathbf{P}'_1$  but  $b \notin \{a_0, \dots, a_{m-1}\}$ . According to C2',  $b$  occurs  $g = n + 1$  times in  $\mathbf{P}'$ . By C3-b or (6), there are at least  $n$  “\*”s in row  $j$  of  $\mathbf{P}'_1$ . Together with the “\*” in the first column, there are at least  $n + 1$  “\*”s in row  $j$  of  $\mathbf{P}'$ , which contradicts C4. Therefore, we have

$$\mathcal{P}'_1 \subseteq \{a_1, \dots, a_m\}$$

By means of (14), (15), and the above equation, we arrive at

$$\mathcal{P}'_1 = \{a_1, \dots, a_m\} \text{ and } \mathcal{P}'_1 \cap \mathcal{P}'_2 = \emptyset$$

In addition, it follows from the above discussion that

- Each integer  $a_j$  appears  $n$  times in  $\mathbf{P}'_1$  since it occurs  $n + 1$  times in  $\mathbf{P}'$ ;
- “\*” appears  $n - 1$  times in each row of  $\mathbf{P}'_1$  since it occurs  $n$  times in each row of  $\mathbf{P}'$ .

This is to say, the  $F_1 \times (K - 1)$  array  $\mathbf{P}'_1$  satisfies C2' and C4. Moreover,  $\mathbf{P}'$  satisfies C3, so does its sub-array  $\mathbf{P}'_1$ . Then applying the assumption in (12), we obtain

$$F_1 \geq \binom{K-1}{n-1} \quad (16)$$

Finally, combining (13) and (16), we have

$$\begin{aligned} F &\geq \frac{F_1 K}{n} \\ &\geq \frac{K}{n} \cdot \binom{K-1}{n-1} \\ &= \frac{K}{n} \cdot \frac{(K-1)!}{(n-1)! (K-n)!} \\ &= \binom{K}{n} \end{aligned}$$

That is, the claim is true for  $g = n + 1$ , which finishes the proof. ■

Based on Lemmas 1 and 2, we are able to prove the following theorem.

**Theorem 3.** For a  $g$ -( $K, F, Z, S$ ) PDA  $\mathbf{P}$ , if  $g = KZ/F + 1 \geq 2$ , then  $F \geq \binom{K}{KZ/F}$ .

*Proof:* By Lemma 1, PDA  $\mathbf{P}$  satisfies C4. Then, the desired result follows directly from Lemma 2. ■

Applying Lemma 1 and Theorem 3 to a  $g$ -( $K, F, Z, S$ ) PDA in place of  $Z/F = M/N$ , we have

$$R = \frac{K(1 - \frac{M}{N})}{g} \quad (17)$$

$$g \leq \frac{KM}{N} + 1 \quad (18)$$

$$F \geq \binom{K}{\frac{KM}{N}} \quad (19)$$

Recall that the minimal rate  $R$  is the primary concern of a  $(K, M, N)$  caching system. According to (17), it is equivalent to maximizing the coding gain  $g$ , which is however upper bounded by (18). As for Ali-Niesen scheme, we see from Theorem 2 that

$$g_{A-N} = \frac{KM}{N} + 1, \quad F_{A-N} = \binom{K}{\frac{KM}{N}}$$

In this sense, Ali-Niesen scheme is optimal since it achieves the maximal coding gain with the least  $F$ .

## V. A NEW CONSTRUCTION OF PDA

Roughly speaking, (18) states that the maximal coding gain of a regular PDA can not exceed  $KM/N + 1$ . Further, (19) tells us that to achieve the maximal coding gain,  $F$  has to be at least  $\binom{K}{KM/N}$ , which increases approximately as same as  $(\frac{N}{M})^{K\frac{M}{N}}(\frac{N}{N-M})^{K(1-\frac{M}{N})}$ . Naturally, it is very desirable if we can decrease  $F$  dramatically with a cost of a slight decrease of coding gain. In this section, we consider this for both small cache size and large cache size. For small cache size, we focus the case that  $N$  is a multiple of  $M$ , *i.e.*  $M/N = 1/q$  for some integer  $q \geq 2$ . Symmetrically, for large cache size, we consider the case  $M/N = (q-1)/q$  for some  $q \geq 2$ .

Our construction can be conveniently described by  $q$ -ary representation. Given an integer  $r \in [0, q^m)$  where  $m \in \mathbb{N}^+$ , with  $r = \sum_{l=0}^{m-1} r_l q^l$  for integers  $r_l \in [0, q)$ , we refer to  $r = (r_{m-1}, \dots, r_0)_q$  as the  $q$ -ary representation of  $r$ .

### A. A New Construction for $M/N = 1/q$

Let  $0 \leq j = (j_{m-1}, \dots, j_0)_q < q^m$  and  $0 \leq k = k_1 q + k_0 < q(m+1)$  where  $m \in \mathbb{N}^+$ ,  $0 \leq j_{m-1}, \dots, j_0 < q$ ,  $0 \leq k_1 \leq m$ , and  $0 \leq k_0 < q$ . Define the entry in row  $j$  and column  $k$  in a  $q^m \times q(m+1)$  array  $\mathbf{A}^{q,m}$  by

- $0 \leq k_1 < m$

$$a_{j,k} = \begin{cases} * & \text{if } j_{k_1} = k_0 \\ (j_{k_1} - k_0 - 1, j_{m-1}, \dots, j_{k_1+1}, k_0, j_{k_1-1}, \dots, j_0)_q & \text{if } j_{k_1} \neq k_0 \end{cases} \quad (20)$$

- $k_1 = m$

$$a_{j,k} = \begin{cases} * & \text{if } j_0 + \dots + j_{m-1} = k_0 \\ (k_0 - \sum_{l=0}^m j_l - 1, j_{m-1}, \dots, j_0)_q & \text{if } j_0 + \dots + j_{m-1} \neq k_0 \end{cases} \quad (21)$$

where all the additions and subtractions are performed modulo  $q$ .

**Theorem 4.** Given  $q, m \in \mathbb{N}^+$ ,  $q \geq 2$ , the array  $\mathbf{A}^{q,m}$  given in (20) and (21) is an  $(m+1)$ -( $q(m+1)$ ,  $q^m$ ,  $q^{m-1}$ ,  $q^{m+1} - q^m$ ) PDA with rate  $R = q - 1$ .

*Proof:* It is sufficient to verify C1, C2', and C3. First, C1 is obvious. As for the other two, note from (20) and (21) that for any integer  $s \in [0, q^{m+1} - q^m)$ , *i.e.*,  $s = (s_m, s_{m-1}, \dots, s_0)_q$  with  $0 \leq s_m < q - 1$ , appears in the entries in row  $j$  and column  $k = k_1 q + k_0$ ,  $0 \leq k_1 \leq m$  and  $0 \leq k_0 < q$ , if and only if

- when  $0 \leq k_1 < m$ ,

$$j = (s_{m-1}, \dots, s_{k_1+1}, s_{k_1} + s_m + 1, s_{k_1-1}, \dots, s_0)_q \quad (22)$$

$$k_0 = s_{k_1} \quad (23)$$

- when  $k_1 = m$ ,

$$j = (s_{m-1}, \dots, s_0)_q \quad (24)$$

$$k_0 = s_0 + \dots + s_{m-1} + s_m + 1 \quad (25)$$

where all the additions are performed modulo  $q$ . Based on the above observation, C2' is clear since for any fixed  $k_1$ , there is exactly one  $j$  and  $k_0$ , such that  $p_{j,k} = s$ , thus each integer occurs once for each possible value of  $k_1$ , *i.e.* each integer  $j \in [0, q^{m+1} - q^m)$  occurs  $m+1$  times in the array  $\mathbf{A}^{q,m}$ .

Further, the observation indicates that for any two distinct entries  $a_{j',k'}$  and  $a_{j'',k''}$ ,  $0 \leq j', j'' < q^m$  and  $0 \leq k' = k'_1 q + k'_0$ ,  $k'' = k''_1 q + k''_0 < q(m+1)$  with  $0 \leq k'_1, k''_1 \leq m$  and  $0 \leq k'_0, k''_0 < q$ , if  $a_{j',k'} = a_{j'',k''} = s \in [0, q^{m+1} - q^m)$ , then

$$k'_1 \neq k''_1$$

since each integer occurs once for each possible  $k_1$ . Moreover, by (22) and (25),

$$j' \neq j''$$

i.e. they lie in different rows and columns. W.L.O.G., assuming that  $k'_1 < k''_1$ , we have  $a_{j',k''} = a_{j'',k'} = *$  from (20) and (21) since

- if  $0 \leq k'_1 < k''_1 < m$ ,  $j'_{k'_1} = s_{k'_1} = k'_0$  and  $j''_{k'_1} = s_{k'_1} = k'_0$  by (22) and (23);
- if  $0 \leq k'_1 < k''_1 = m$ ,  $j'_0 + \dots + j'_{m-1} = s_0 + \dots + s_{m-1} + s_m + 1 = k'_0$  and  $j''_{k'_1} = s_{k'_1} = k'_0$  by (22)-(25).

That is, C3 holds. This finishes the proof. ■

*Example 4.* The array in (7) in Example 2 is in fact generated by setting  $q = 2, m = 2$ . Table II presents the binary representation form of (20)-(21). Clearly, the corresponding  $4 \times 6$  array is  $\mathbf{A}^{2,2}$  in (7).

TABLE II: Binary representation of  $\mathbf{A}^{2,2}$  with  $m = 2$  and  $q = 2$

| $(j_1, j_0)_2 \backslash (k_1, k_0)$ | (0, 0)        | (0, 1)        | (1, 0)        | (1, 1)        | (2, 0)        | (2, 1)        |
|--------------------------------------|---------------|---------------|---------------|---------------|---------------|---------------|
| $(0, 0)_2$                           | *             | $(0, 0, 1)_2$ | *             | $(0, 1, 0)_2$ | *             | $(0, 0, 0)_2$ |
| $(0, 1)_2$                           | $(0, 0, 0)_2$ | *             | *             | $(0, 1, 1)_2$ | $(0, 0, 1)_2$ | *             |
| $(1, 0)_2$                           | *             | $(0, 1, 1)_2$ | $(0, 0, 0)_2$ | *             | $(0, 1, 0)_2$ | *             |
| $(1, 1)_2$                           | $(0, 1, 0)_2$ | *             | $(0, 0, 1)_2$ | *             | *             | $(0, 1, 1)_2$ |

*Example 5.* Let  $q = 3, m = 2$ , then Table III gives the ternary representation of the new construction. Then, the corresponding

TABLE III: Ternary representation of  $\mathbf{A}^{3,2}$  with  $m = 2$  and  $q = 3$

| $(j_1, j_0)_3 \backslash (k_1, k_0)$ | (0, 0)        | (0, 1)        | (0, 2)        | (1, 0)        | (1, 1)        | (1, 2)        | (2, 0)        | (2, 1)        | (2, 2)        |
|--------------------------------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| $(0, 0)_3$                           | *             | $(1, 0, 1)_3$ | $(0, 0, 2)_3$ | *             | $(1, 1, 0)_3$ | $(0, 2, 0)_3$ | *             | $(0, 0, 0)_3$ | $(1, 0, 0)_3$ |
| $(0, 1)_3$                           | $(0, 0, 0)_3$ | *             | $(1, 0, 2)_3$ | *             | $(1, 1, 1)_3$ | $(0, 2, 1)_3$ | $(1, 0, 1)_3$ | *             | $(0, 0, 1)_3$ |
| $(0, 2)_3$                           | $(1, 0, 0)_3$ | $(0, 0, 1)_3$ | *             | *             | $(1, 1, 2)_3$ | $(0, 2, 2)_3$ | $(0, 0, 2)_3$ | $(1, 0, 2)_3$ | *             |
| $(1, 0)_3$                           | *             | $(1, 1, 1)_3$ | $(0, 1, 2)_3$ | $(0, 0, 0)_3$ | *             | $(1, 2, 0)_3$ | $(1, 1, 0)_3$ | *             | $(0, 1, 0)_3$ |
| $(1, 1)_3$                           | $(0, 1, 0)_3$ | *             | $(1, 1, 2)_3$ | $(0, 0, 1)_3$ | *             | $(1, 2, 1)_3$ | $(0, 1, 1)_3$ | $(1, 1, 1)_3$ | *             |
| $(1, 2)_3$                           | $(1, 1, 0)_3$ | $(0, 1, 1)_3$ | *             | $(0, 0, 2)_3$ | *             | $(1, 2, 2)_3$ | *             | $(0, 1, 2)_3$ | $(1, 1, 2)_3$ |
| $(2, 0)_3$                           | *             | $(1, 2, 1)_3$ | $(0, 2, 2)_3$ | $(1, 0, 0)_3$ | $(0, 1, 0)_3$ | *             | $(0, 2, 0)_3$ | $(1, 2, 0)_3$ | *             |
| $(2, 1)_3$                           | $(0, 2, 0)_3$ | *             | $(1, 2, 2)_3$ | $(1, 0, 1)_3$ | $(0, 1, 1)_3$ | *             | *             | $(0, 2, 1)_3$ | $(1, 2, 1)_3$ |
| $(2, 2)_3$                           | $(1, 2, 0)_3$ | $(0, 2, 1)_3$ | *             | $(1, 0, 2)_3$ | $(0, 1, 2)_3$ | *             | $(1, 2, 2)_3$ | *             | $(0, 2, 2)_3$ |

$9 \times 9$  array is

$$\mathbf{A}^{3,2} = \begin{bmatrix} * & 10 & 2 & * & 12 & 6 & * & 0 & 9 \\ 0 & * & 11 & * & 13 & 7 & 10 & * & 1 \\ 9 & 1 & * & * & 14 & 8 & 2 & 11 & * \\ * & 13 & 5 & 0 & * & 15 & 12 & * & 3 \\ 3 & * & 14 & 1 & * & 16 & 4 & 13 & * \\ 12 & 4 & * & 2 & * & 17 & * & 5 & 14 \\ * & 16 & 8 & 9 & 3 & * & 6 & 15 & * \\ 6 & * & 17 & 10 & 4 & * & * & 7 & 16 \\ 15 & 7 & * & 11 & 5 & * & 17 & * & 8 \end{bmatrix}$$

Comparing  $\mathbf{A}^{2,2}$  in (7) and  $\mathbf{D}^{6,3}$  in (9), they both support 6 users with  $M/N = 1/2$ . While having 1 less coding gain (3 versus 4) and thus a larger rate (1 versus  $3/4$ ) compared to  $\mathbf{D}^{6,3}$ ,  $\mathbf{A}^{2,2}$  has the advantage that the number of packets needed to be split into is much smaller (4 versus 20).

### B. A New Construction for $M/N = (q-1)/q$

Let  $0 \leq j = (j_m, j_{m-1}, \dots, j_0)_q < (q-1)q^m$  and  $0 \leq k = k_1q + k_0 < q(m+1)$  where  $m \in \mathbb{N}^+$ ,  $0 \leq j_{m-1}, \dots, j_0 < q$ ,  $0 \leq j_m < q-1$ ,  $0 \leq k_1 \leq m$ , and  $0 \leq k_0 < q$ . Define the entry in row  $j$  and column  $k$  in a  $(q-1)q^m \times q(m+1)$  array  $\mathbf{B}^{q,m}$  by

- $0 \leq k_1 < m$

$$b_{j,k} = \begin{cases} (j_{m-1}, \dots, j_{k_1+1}, j_{k_1} + j_m + 1, j_{k_1-1}, \dots, j_0)_q & \text{if } j_{k_1} = k_0 \\ * & \text{if } j_{k_1} \neq k_0 \end{cases} \quad (26)$$

- $k_1 = m$

$$b_{j,k} = \begin{cases} (j_{m-1}, j_{m-2}, \dots, j_0)_q & \text{if } j_0 + \dots + j_{m-1} + j_m + 1 = k_0 \\ * & \text{if } j_0 + \dots + j_{m-1} + j_m + 1 \neq k_0 \end{cases} \quad (27)$$

where all the additions are performed modulo  $q$ .

**Theorem 5.** Given  $q, m \in \mathbb{N}^+$ ,  $q \geq 2$ , the array  $\mathbf{B}^{q,m}$  given in (26) and (27) is an  $(q-1) \cdot (m+1) - (q(m+1), (q-1)q^m, (q-1)^2q^{m-1}, q^m)$  PDA with rate  $R = 1/(q-1)$ .

*Proof:* Similar to the proof of Theorem 4, it is sufficient to verify C1, C2', and C3. First, C1 is obvious. As for the other two, note from (26) and (27) that for any integer  $s \in [0, q^m)$ , i.e.,  $s = (s_{m-1}, \dots, s_0)_q$  with  $0 \leq s_l < q$  for  $0 \leq l < m$ , appears in the entries in row  $j$  and column  $k = k_1q + k_0$ ,  $0 \leq k_1 \leq m$  and  $0 \leq k_0 < q$ , if and only if

- when  $0 \leq k_1 < m$ ,

$$j = (j_m, s_{m-1}, \dots, s_{k_1+1}, s_{k_1} - j_m - 1, s_{k_1-1}, \dots, s_0)_q \quad (28)$$

$$k_0 = s_{k_1} - j_m - 1 \quad (29)$$

- when  $k_1 = m$ ,

$$j = (j_m, s_{m-1}, \dots, s_0)_q \quad (30)$$

$$k_0 = s_0 + \dots + s_{m-1} + j_m + 1 \quad (31)$$

where all the additions are performed modulo  $q$ . Based on the above observation, C2' is clear since for any fixed  $k_1$  and  $j_m$ , there is exactly one  $k_0$ , such that  $p_{j,k} = s$ , thus each integer occurs once for each possible value of  $k_1, j_m$ , i.e. each integer  $s \in [0, q^m)$  occurs  $(q-1)(m+1)$  times in the array  $\mathbf{B}^{q,m}$ .

Further, we claim that for any two distinct entries  $b_{j',k'}$  and  $b_{j'',k''}$ ,  $0 \leq j', j'' < (q-1)q^m$  and  $0 \leq k' = k'_1q + k'_0$ ,  $k'' = k''_1q + k''_0 < q(m+1)$  with  $0 \leq k'_1, k''_1 \leq m$  and  $0 \leq k'_0, k''_0 < q$ , if  $b_{j',k'} = b_{j'',k''} = s \in [0, q^m)$ , then

$$k' \neq k'', \quad j' \neq j''$$

i.e. they lie in different rows and columns. This is true because

- if  $j' \neq j''$ , assume  $k' = k''$ , then  $k'_1 = k''_1$ ,  $k'_0 = k''_0$ . Then by (28) (or (30)), and fact  $j' \neq j''$ ,  $j'_m \neq j''_m$ . Therefore, by (29) (or (31)),  $k'_0 \neq k''_0$ , contradiction.
- if  $k' \neq k''$ , assume  $j' = j''$ . W.L.O.G, assume  $0 \leq k' < k'' < q(m+1)$ , then  $k'_1 \leq k''_1$ .  
1) if  $k'_1 = k''_1$ , then since  $j'_m = j''_m$ , by (29) (or (31)),

$$k'_0 = k''_0$$

thus,  $k' = k''$ , contradiction with  $k' \neq k''$ .

- 2) if  $0 \leq k'_1 < k''_1 \leq m$ , then by (28) (and (30) in case  $k''_1 = m$ ) and the fact  $j'_m = j''_m$ , we have

$$j'_m + 1 = 0 \pmod{q}$$

which indicates  $j'_m \pmod{q} = q-1$ . This contradicts with the fact  $0 \leq j'_m < q-1$ .

Therefore,  $s$  actually lies in different rows and different columns. W.L.O.G., assuming that  $k' < k''$ , then  $k'_1 \leq k''_1$ . We have  $b_{j',k''} = b_{j'',k'} = *$  from (26) and (27) since

- if  $k'_1 = k''_1$ , then  $k'_0 \neq k''_0$  by the fact  $k' \neq k''$ .

1) if  $0 \leq k'_1 = k''_1 < m$ , by (26),

$$j'_{k'_1} = j'_{k'_1} = k'_0 \neq k''_0$$

$$j''_{k'_1} = j''_{k'_1} = k''_0 \neq k'_0$$

2) if  $k'_1 = k''_1 = m$ , by (27),

$$j'_0 + j'_1 + \cdots + j'_m + 1 = k'_0 \neq k''_0$$

$$j''_0 + j''_1 + \cdots + j''_m + 1 = k''_0 \neq k'_0$$

- if  $0 \leq k'_1 < k''_1 < m$ ,  $j'_{k'_1} = s_{k'_1} \neq k''_0$  and  $j''_{k'_1} = s_{k'_1} \neq k'_0$  by (28) and (29) and the fact that both  $j'_m + 1$  and  $j''_m + 1$  are not 0;
- if  $0 \leq k'_1 < k''_1 = m$ ,  $j'_0 + \cdots + j'_{m-1} + j'_m + 1 = s_0 + \cdots + s_{m-1} \neq k''_0$  and  $j''_{k'_1} = s_{k'_1} \neq k'_0$  by (28)-(31) and the fact that both  $j'_m + 1$  and  $j''_m + 1$  are not 0.

That is, C3 holds. This finishes the proof. ■

*Example 6.* Let  $q = 3, m = 2$ , then Table IV gives the ternary representation of the new construction. Then, the corresponding

TABLE IV: Ternary representation of  $\mathbf{B}^{3,2}$  with  $m = 2$  and  $q = 3$

| $(j_2, j_1, j_0)_3 \setminus (k_1, k_0)$ | (0, 0)              | (0, 1)              | (0, 2)              | (1, 0)              | (1, 1)              | (1, 2)              | (2, 0)              | (2, 1)              | (2, 2)               |
|--|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|---------------------|----------------------|
| (0, 0, 0) <sub>3</sub>                   | (0, 1) <sub>3</sub> | *                   | *                   | (1, 0) <sub>3</sub> | *                   | *                   | *                   | (0, 0) <sub>3</sub> | *                    |
| (0, 0, 1) <sub>3</sub>                   | *                   | (0, 2) <sub>3</sub> | *                   | (1, 1) <sub>3</sub> | *                   | *                   | *                   | *                   | (0, 1) <sub>3</sub>  |
| (0, 0, 2) <sub>3</sub>                   | *                   | *                   | (0, 0) <sub>3</sub> | (1, 2) <sub>3</sub> | *                   | *                   | (0, 2) <sub>3</sub> | *                   | *                    |
| (0, 1, 0) <sub>3</sub>                   | (1, 1) <sub>3</sub> | *                   | *                   | *                   | (2, 0) <sub>3</sub> | *                   | *                   | *                   | (1, 0) <sub>3</sub>  |
| (0, 1, 1) <sub>3</sub>                   | *                   | (1, 2) <sub>3</sub> | *                   | *                   | (2, 1) <sub>3</sub> | *                   | (1, 1) <sub>3</sub> | *                   | *                    |
| (0, 1, 2) <sub>3</sub>                   | *                   | *                   | (1, 0) <sub>3</sub> | *                   | (2, 2) <sub>3</sub> | *                   | *                   | (1, 2) <sub>3</sub> | *                    |
| (0, 2, 0) <sub>3</sub>                   | (2, 1) <sub>3</sub> | *                   | *                   | *                   | *                   | (0, 0) <sub>3</sub> | (2, 0) <sub>3</sub> | *                   | *                    |
| (0, 2, 1) <sub>3</sub>                   | *                   | (2, 2) <sub>3</sub> | *                   | *                   | *                   | (0, 1) <sub>3</sub> | *                   | (2, 1) <sub>3</sub> | *                    |
| (0, 2, 2) <sub>3</sub>                   | *                   | *                   | (2, 0) <sub>3</sub> | *                   | *                   | (0, 2) <sub>3</sub> | *                   | *                   | (2, 2) <sub>3</sub>  |
| (1, 0, 0) <sub>3</sub>                   | (0, 2) <sub>3</sub> | *                   | *                   | (2, 0) <sub>3</sub> | *                   | *                   | *                   | *                   | (0, 0) <sub>3</sub>  |
| (1, 0, 1) <sub>3</sub>                   | *                   | (0, 0) <sub>3</sub> | *                   | (2, 1) <sub>3</sub> | *                   | *                   | (0, 1) <sub>3</sub> | *                   | *                    |
| (1, 0, 2) <sub>3</sub>                   | *                   | *                   | (0, 1) <sub>3</sub> | (2, 2) <sub>3</sub> | *                   | *                   | *                   | (0, 2) <sub>3</sub> | *                    |
| (1, 1, 0) <sub>3</sub>                   | (1, 2) <sub>3</sub> | *                   | *                   | *                   | (0, 0) <sub>3</sub> | *                   | (1, 0) <sub>3</sub> | *                   | *                    |
| (1, 1, 1) <sub>3</sub>                   | *                   | (1, 0) <sub>3</sub> | *                   | *                   | (0, 1) <sub>3</sub> | *                   | *                   | (1, 1) <sub>3</sub> | *                    |
| (1, 1, 2) <sub>3</sub>                   | *                   | *                   | (1, 1) <sub>3</sub> | *                   | (0, 2) <sub>3</sub> | *                   | *                   | *                   | *(1, 2) <sub>3</sub> |
| (1, 2, 0) <sub>3</sub>                   | (2, 2) <sub>3</sub> | *                   | *                   | *                   | *                   | (1, 0) <sub>3</sub> | *                   | (2, 0) <sub>3</sub> | *                    |
| (1, 2, 1) <sub>3</sub>                   | *                   | (2, 0) <sub>3</sub> | *                   | *                   | *                   | (1, 1) <sub>3</sub> | *                   | *                   | (2, 1) <sub>3</sub>  |
| (1, 2, 2) <sub>3</sub>                   | *                   | *                   | (2, 1) <sub>3</sub> | *                   | *                   | (1, 2) <sub>3</sub> | (2, 2) <sub>3</sub> | *                   | *                    |

$9 \times 18$  array is

$$\mathbf{B}^{3,2} = \begin{bmatrix} 1 & * & * & 3 & * & * & * & 0 & * \\ * & 2 & * & 4 & * & * & * & * & 1 \\ * & * & 0 & 5 & * & * & 2 & * & * \\ 4 & * & * & * & 6 & * & * & * & 3 \\ * & 5 & * & * & 7 & * & 4 & * & * \\ * & * & 3 & * & 8 & * & * & 5 & * \\ 7 & * & * & * & * & 0 & 6 & * & * \\ * & 8 & * & * & * & 1 & * & 7 & * \\ * & * & 6 & * & * & 2 & * & * & 8 \\ 2 & * & * & 6 & * & * & * & * & 0 \\ * & 0 & * & 7 & * & * & 1 & * & * \\ * & * & 1 & 8 & * & * & * & 2 & * \\ 5 & * & * & * & 0 & * & 3 & * & * \\ * & 3 & * & * & 1 & * & * & 4 & * \\ * & * & 4 & * & 2 & * & * & * & 5 \\ 8 & * & * & * & * & 3 & * & 6 & * \\ * & 6 & * & * & * & 4 & * & * & 7 \\ * & * & 7 & * & * & 5 & 8 & * & * \end{bmatrix}$$

*Remark 1.* Theorem 4 (or Theorem 5 respectively) shows that we are able to construct a  $F \times K = q^m \times q(m+1)$  (or  $(q-1)q^m \times q(m+1)$ ) PDA, which supports  $K = q(m+1)$  users when  $M/N = 1/q$  (or  $(q-1)/q$ ). For a general properly large  $K$  ( $K \geq 2q$ ), one can construct a PDA to support  $K$  users by setting  $m = \lceil \frac{K}{q} \rceil - 1$ , and construct a  $q^{\lceil \frac{K}{q} \rceil - 1} \times q^{\lceil \frac{K}{q} \rceil}$  (or  $(q-1)q^{\lceil \frac{K}{q} \rceil - 1} \times q^{\lceil \frac{K}{q} \rceil}$ ) PDA firstly, and then delete any  $q^{\lceil \frac{K}{q} \rceil} - K$  columns, the resultant rate is not larger than  $q-1$  (or  $1/(q-1)$ ).

*Remark 2.* Note that, if  $1/q = 1 - 1/q$ , i.e.,  $q = 2$ , for any given  $m \in \mathbb{N}^+$ , both Theorem 4 and Theorem 5 result a  $(m+1) - (2(m+1), 2^m, 2^{m-1}, 2^m)$  PDA. Interestingly, they are equivalent to each other with respect to column permutation. For example, when  $m = 2$ , Theorem 4 gives  $\mathbf{A}^{2,2}$  in (7). While Theorem 5 gives

$$\mathbf{B}^{2,2} = \begin{bmatrix} 1 & * & 2 & * & * & 0 \\ * & 0 & 3 & * & 1 & * \\ 3 & * & * & 0 & 2 & * \\ * & 2 & * & 1 & * & 3 \end{bmatrix}$$

Obviously, if we exchange the columns 1 and 3 with columns 0 and 2 of  $\mathbf{A}^{2,2}$  respectively, we will obtain  $\mathbf{B}^{2,2}$ .

## VI. COMPARISON WITH ALI-NIESEN SCHEME

In this section, we compare the proposed scheme with Ali-Niesen scheme by setting  $M/N = 1/q$  and  $(q-1)/q$  respectively,  $K = q(m+1)$  with  $q, m \in \mathbb{N}^+$ ,  $q \geq 2$ .

It is easy to observe that, when  $K = q(m+1)$ ,  $q, m \in \mathbb{N}^+$ , when  $M/N = 1/q$  or  $(q-1)/q$ , the coding gain achieved by Ali-Niesen scheme and ours are

$$g_{A-N} = \frac{KM}{N} + 1 \quad \text{and} \quad g_{New} = \frac{KM}{N}$$

respectively. That is, there is only 1 loss in coding gain, i.e.,

$$g_{A-N} - g_{New} = 1 \tag{32}$$

Consequently, the rates of Ali-Niesen scheme and ours are respectively

$$R_{A-N} = \frac{K(1 - \frac{M}{N})}{g_{A-N}} \quad \text{and} \quad R_{New} = \frac{K(1 - \frac{M}{N})}{g_{New}}$$

Define

$$\lambda_{K, \frac{M}{N}} \triangleq \frac{R_{A-N}}{R_{New}} = \frac{KM/N}{KM/N + 1}$$

While on the other hand, Ali-Niesen scheme and ours have to split each file into  $F_{A-N}$  packets and  $F_{New}$  packets respectively, where

$$F_{A-N} = \binom{K}{\frac{KM}{N}} = \binom{K}{\frac{K}{q}} \quad \text{if} \quad \frac{M}{N} = \frac{1}{q}, \quad \frac{q-1}{q} \quad (33)$$

$$F_{New} = \begin{cases} q^{\frac{K}{q}-1} & \text{if } \frac{M}{N} = \frac{1}{q} \\ (q-1)q^{\frac{K}{q}-1} & \text{if } \frac{M}{N} = \frac{q-1}{q} \end{cases} \quad (34)$$

Define

$$\eta_{K, \frac{M}{N}} \triangleq \frac{F_{A-N}}{F_{New}}$$

Clearly,

$$\lim_{K \rightarrow \infty} \lambda_{K, \frac{M}{N}} = 1 \quad (35)$$

Regarding the value of  $\eta_{K, \frac{M}{N}}$ , we derive the following theorem.

**Theorem 6.** As  $K \rightarrow \infty$ , for  $M/N = 1/q$ ,

$$\eta_{K, \frac{M}{N}} \sim \frac{q^2}{\sqrt{2\pi K(q-1)}} \cdot \left(\frac{q}{q-1}\right)^{K(1-\frac{1}{q})} \quad (36)$$

For  $M/N = (q-1)/q$ ,

$$\eta_{K, \frac{M}{N}} \sim \frac{q^2}{\sqrt{2\pi K(q-1)^{\frac{3}{2}}}} \cdot \left(\frac{q}{q-1}\right)^{K(1-\frac{1}{q})} \quad (37)$$

*Proof:* The well-known Stirling's formula tells us

$$n! \sim \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

as  $n \rightarrow \infty$  for  $n \in \mathbb{N}^+$ . Therefore, when  $K \rightarrow \infty$ , we have

$$\begin{aligned} \binom{K}{\frac{K}{q}} &= \frac{K!}{\frac{K}{q}!(K - \frac{K}{q})!} \\ &\sim \frac{\sqrt{2\pi K} \left(\frac{K}{e}\right)^K}{\sqrt{2\pi \frac{K}{q}} \left(\frac{K}{qe}\right)^{\frac{K}{q}} \sqrt{2\pi K(1 - \frac{1}{q})} \left(\frac{K}{e}(1 - \frac{1}{q})\right)^{K(1-\frac{1}{q})}} \\ &= \frac{1}{\sqrt{2\pi K \frac{1}{q}(1 - \frac{1}{q})}} \cdot \frac{1}{\left(\frac{1}{q}\right)^{\frac{K}{q}}} \cdot \frac{1}{(1 - \frac{1}{q})^{K(1-\frac{1}{q})}} \\ &= \frac{q}{\sqrt{2\pi K(q-1)}} \cdot q^{\frac{K}{q}} \cdot \left(\frac{q}{q-1}\right)^{K(1-\frac{1}{q})} \end{aligned}$$

Then, for  $M/N = 1/q$ ,

$$\begin{aligned} \eta_{K, \frac{M}{N}} &= \frac{\binom{K}{\frac{K}{q}}}{q^{\frac{K}{q}-1}} \\ &\sim \frac{q^2}{\sqrt{2\pi K(q-1)}} \cdot \left(\frac{q}{q-1}\right)^{K(1-\frac{1}{q})} \end{aligned}$$



For  $M/N = (q-1)/q$ ,

$$\begin{aligned}\eta_{K, \frac{M}{N}} &= \frac{\binom{K}{\frac{K}{q}}}{(q-1)q^{\frac{K}{q}-1}} \\ &\sim \frac{q^2}{\sqrt{2\pi K}(q-1)^{\frac{3}{2}}} \cdot \left(\frac{q}{q-1}\right)^{K(1-\frac{1}{q})}\end{aligned}$$

■

According to (32) and (35), compared to Ali-Niesen scheme, our new scheme achieves a coding gain 1 less than Ali-Niesen scheme, or in terms of rate, the ratio  $\lambda_{K, \frac{M}{N}}$  is approximately 1 when  $K$  becomes large. While from (33), (34) and Theorem 6, it is clear that,  $F_{A-N}$  is of order  $O\left(e^{K \cdot \left(\frac{1}{q} \ln q + (1-\frac{1}{q}) \ln \frac{q}{q-1}\right)}\right)$  and  $F_{New}$  is of order  $O\left(e^{K \cdot \frac{1}{q} \ln q}\right)$ . The new construction saves a factor  $\eta_{K, \frac{M}{N}}$  of order  $O\left(e^{K \cdot (1-\frac{1}{q}) \ln \frac{q}{q-1}}\right)$ , which goes to infinity exponentially with  $K$ . Finally, we summary the comparisons in Table V.

TABLE V: Performance comparison of Ali-Niesen scheme and new scheme

|                               | Performance | Ali-Niesen scheme   | New scheme               |
|-------------------------------|-------------|---|--------------------------|
| $\frac{M}{N} = \frac{1}{q}$   | $g$         | $\frac{K}{q} + 1$   | $\frac{K}{q}$            |
|                               | $R$         | $\frac{K}{K+q}(q-1)$  | $q-1$                    |
|                               | $F$         | $\sim \frac{q}{\sqrt{2\pi K(q-1)}} \cdot q^{\frac{K}{q}} \cdot \left(\frac{q}{q-1}\right)^{K(1-\frac{1}{q})}$ | $q^{\frac{K}{q}-1}$      |
| $\frac{M}{N} = \frac{q-1}{q}$ | $g$         | $K \frac{q-1}{q} + 1$   | $K \frac{q-1}{q}$        |
|                               | $R$         | $\frac{K}{q+K(q-1)}$  | $\frac{1}{q-1}$          |
|                               | $F$         | $\sim \frac{q}{\sqrt{2\pi K(q-1)}} \cdot q^{\frac{K}{q}} \cdot \left(\frac{q}{q-1}\right)^{K(1-\frac{1}{q})}$ | $(q-1)q^{\frac{K}{q}-1}$ |

Table VI presents some numerical comparison results for  $M/N = 1/3, 1/2$  and  $2/3$  for some  $K$ .

## VII. CONCLUSIONS

In this paper, we defined a new array PDA, which can be used to describe the placement and delivery schemes in caching system, for example Ali-Niesen caching scheme. Based on a PDA of size  $F \times K$ , the caching scheme can support  $K$  users by dividing each file into  $F$  packets. Therefore, the problem of designing a centralized caching scheme can be translated into a problem of designing an appropriate PDA. Particularly, we established an upper bound on coding gain for all possible regular PDAs and proved that Ali-Niesen scheme achieves the upper bound with the least possible  $F$  in all schemes corresponding to regular PDAs. Furthermore, we presented a new construction of PDA for the cases  $M/N$  is  $1/q$  and  $(q-1)/q$  for each integer  $q \geq 2$ . The new construction leads to less order of  $F$  at the expense of one less coding gain. In terms of rate, the new constructions decrease  $F$  significantly at the expense of a diminishing loss in rate as  $K$  becomes large.

It should be noted that we only focused on centralized network in this paper. In fact, PDA can also be used to describe decentralized networks where the placement is random. Accordingly, it requires that the positions of symbol “\*” are independent of the users and files.

## REFERENCES

- [1] A. Dan, D. Sitaram, and P. Shahabuddin, “Dynamic batching policies for an on-demand video server,” *Multimedia Syst.*, vol. 4, no. 3, pp. 112–121, 1996. [Online]. Available: <http://dx.doi.org/10.1007/s005300050016>

TABLE VI: Numerical Comparisons of Ali-Niesen scheme and new scheme

| $K$                         |           | 6      | 12     | 18     | 24      | 30        | 36         |
|-----------------------------|-----------|--------|--------|--------|---------|-----------|------------|
| $\frac{M}{N} = \frac{1}{3}$ | $g_{A-N}$ | 3      | 5      | 7      | 9       | 11        | 13         |
|                             | $g_{New}$ | 2      | 4      | 6      | 8       | 10        | 12         |
|                             | $R_{A-N}$ | 1.3333 | 1.6000 | 1.7143 | 1.7778  | 1.8182    | 1.8462     |
|                             | $R_{New}$ | 2      | 2      | 2      | 2       | 2         | 2          |
|                             | $F_{A-N}$ | 15     | 495    | 18564  | 735471  | 30045015  | 1251677700 |
|                             | $F_{New}$ | 3      | 27     | 243    | 2187    | 19683     | 177147     |
| $\frac{M}{N} = \frac{1}{2}$ | $g_{A-N}$ | 4      | 7      | 10     | 13      | 16        | 19         |
|                             | $g_{New}$ | 3      | 6      | 9      | 12      | 15        | 18         |
|                             | $R_{A-N}$ | 0.7500 | 0.8571 | 0.9000 | 0.9231  | 0.9375    | 0.9474     |
|                             | $R_{New}$ | 1      | 1      | 1      | 1       | 1         | 1          |
|                             | $F_{A-N}$ | 20     | 924    | 48620  | 2704156 | 155117520 | 9075135300 |
|                             | $F_{New}$ | 4      | 32     | 256    | 2048    | 16384     | 131072     |
| $\frac{M}{N} = \frac{2}{3}$ | $g_{A-N}$ | 5      | 9      | 13     | 17      | 21        | 25         |
|                             | $g_{New}$ | 4      | 8      | 12     | 16      | 20        | 24         |
|                             | $R_{A-N}$ | 0.4000 | 0.4444 | 0.4615 | 0.4706  | 0.4762    | 0.4800     |
|                             | $R_{New}$ | 0.5    | 0.5    | 0.5    | 0.5     | 0.5       | 0.5        |
|                             | $F_{A-N}$ | 15     | 495    | 18564  | 735471  | 30045015  | 1251677700 |
|                             | $F_{New}$ | 6      | 54     | 486    | 4374    | 39366     | 354294     |

- [2] A. Meyerson, K. Munagala, and S. A. Plotkin, "Web caching using access statistics," in *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, 2001, pp. 354–363. [Online]. Available: <http://dl.acm.org/citation.cfm?id=365411.365479>
- [3] Cisco visual networking index: Global mobile data traffic forecast update, 2014-2019. [Online]. Available: <http://goo.gl/1XYhqY>
- [4] I. D. Baev, R. Rajaraman, and C. Swamy, "Approximation algorithms for data placement problems," *SIAM J. Comput.*, vol. 38, no. 4, pp. 1411–1429, 2008. [Online]. Available: <http://dx.doi.org/10.1137/080715421>
- [5] K. C. Almeroth and M. H. Ammar, "The use of multicast delivery to provide a scalable and interactive video-on-demand service," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 6, pp. 1110–1122, 1996. [Online]. Available: <http://dx.doi.org/10.1109/49.508282>
- [6] M. A. Maddah-Ali and U. Niesen, "Decentralized coded caching attains order-optimal memory-rate tradeoff," in *2013 51st Annual Allerton Conference on Communication, Control, and Computing, Allerton Park & Retreat Center, Monticello, IL, USA, October 2-4, 2013*, 2013, pp. 421–427. [Online]. Available: <http://dx.doi.org/10.1109/Allerton.2013.6736555>
- [7] M. A. Maddah-Ali and U. Niesen, "Fundamental limits of caching," *IEEE Transactions on Information Theory*, vol. 60, no. 5, pp. 2856–2867, 2014. [Online]. Available: <http://dx.doi.org/10.1109/TIT.2014.2306938>
- [8] M. Ji, A. M. Tulino, J. Llorca, and G. Caire, "Order-optimal rate of caching and coded multicasting with random demands," *arXiv preprint arXiv:1502.03124*, 2015.
- [9] M. R. Korupolu, C. G. Plaxton, and R. Rajaraman, "Placement algorithms for hierarchical cooperative caching," in *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland.*, 1999, pp. 586–595. [Online]. Available: <http://dl.acm.org/citation.cfm?id=314500.314880>
- [10] N. Golrezaei, A. F. Molisch, A. G. Dimakis, and G. Caire, "Femtocaching and device-to-device collaboration: A new architecture for wireless video distribution," *Communications Magazine, IEEE*, vol. 51, no. 4, pp. 142–149, 2013.
- [11] S. C. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA*, 2010, pp. 1478–1486. [Online]. Available: <http://dx.doi.org/10.1109/INFCOM.2010.5461964>